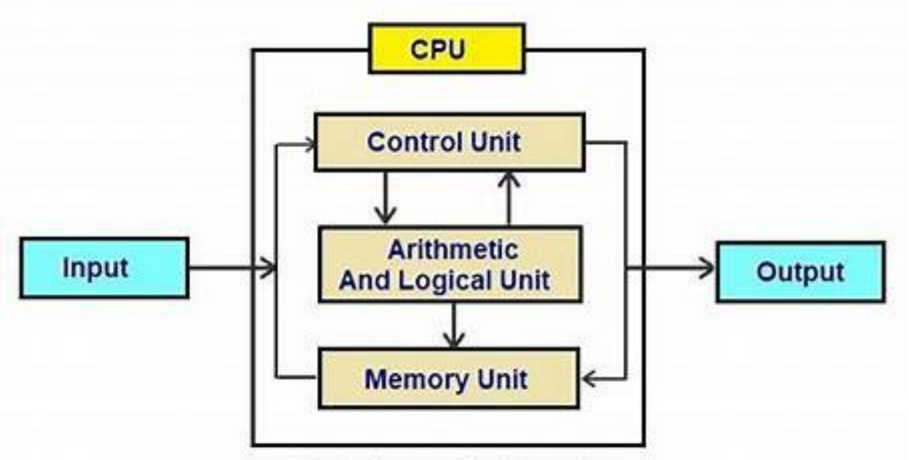
Introduction to Computers

"A Computer is an electronic device which takes input, process it and gives the output".

- The information given to a computer is called "Input".
- The information or result given by the computer is called "Output".



Block Diagram of a Computer

ArtOfTesting

Different basic parts of a computer:

All parts of a computer are divided into the following categories.

1. Input Device
2. Output Device
3. CPU (Central Processing Unit)
4. Primary Storages
5. Auxiliary Storages

1. **Input Devices:**

These are the devices through which input is given to the computer. Ex: Keyboard, Mouse, Scanner, Joystick, Light Pen, Touch Screen, etc.

2. **Output Device:**

These are the devices through which the output is given by the computer. Ex: Monitor, Printer, Speakers etc.

3. **Central Processing Unit (CPU):**

CPU is the main part of a computer system which is responsible for all arithmetic calculations, movement of data, and comparison among the data. It can be called as the Heart of a computer system.

The CPU consists of CU (Control Unit) and ALU (Arithmetic Logic Unit).

CU stores the instruction set, which specifies the operations to be performed by the computer. CU transfers the data and instructions to ALU for Arithmetic operations.

4. **Primary storage:**

The primary storage, also known as main memory, is a place where the programs and data are stored temporarily during processing. The data in primary storage will be erased when we turn off a personal computer (In case of RAM)**.**

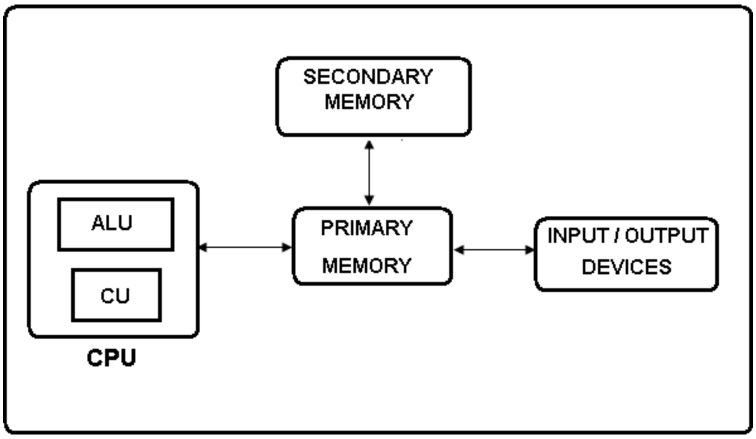
RAM (Random Access Memory):

It is also called the main memory. It is a place where the programs and data are stored temporarily during processing. The data in this memory will be removed, when we switch off the computer.

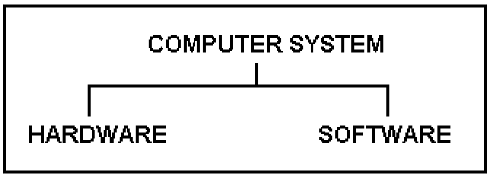
ROM (Read Only Memory):

ROM stores the data even the computer turned off. It is a permanent memory cannot be modified, used to store BIOS (Basic Input Output System).

5. **Auxiliary storage:** Auxiliary storage, also known as secondary storage, is used for both input and output. It is the place where the programs and data are stored permanently. When we turn off the computer, our programs and data remain in the secondary storage, ready for the next time we need them. It is the place where the programs and data stored permanently.



All above are collective called as Hardware.



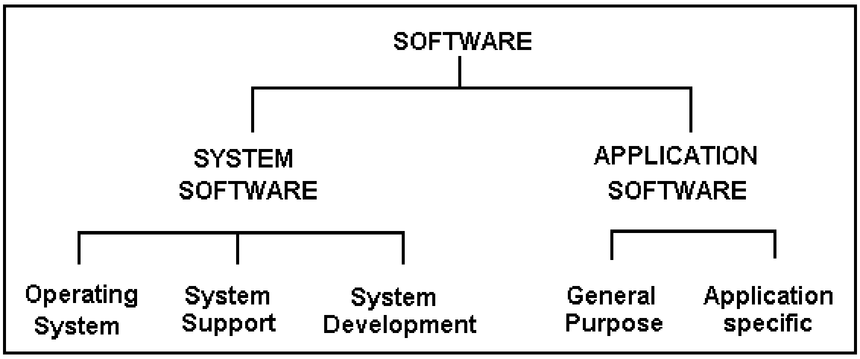
A Computer system consists of two major components.

1. Hardware
2. Software

1. Hardware:
   All physical equipment of a computer system is called Hardware.
2. Software:
   Software is a set of programs that allows Hardware to do specific job.

Software is divided into two categories.

1. System software 2. Application Software



SYSTEM SOFTWARE:

System software manages the computer resources. It provides the interface between the hardware and the users but does nothing to directly serve the users" needs. System software consists of programs that manage the hardware resources of a computer and perform required information processing tasks.
This can be divided into 3 parts.

i. **Operating system:**
The operating system provides services such as a user interface, file and database access, and interfaces to communication systems such as internet protocols. The primary purpose of this software is to keep the system operating in an efficient manner while allowing the users access to the system. It is software which interacts with the hardware directly.

ii. **System support:**
This provides system utilities such as disc format, sort programs etc and operating services such as performance statistics of a computer and security monitors which monitors which protect the system and data.

iii. **System development software:**
It consists of translators which convert high level languages into machine level languages and debuggers, and computer assisted software engineering (CASE) systems, which makes programs error free.

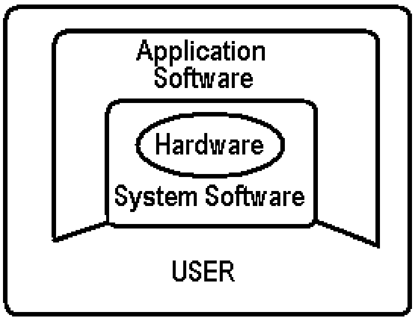
APPLICATION SOFTWARE:
It is the software which is used to solve different problems. It can be divided into two categories.

i. General purpose ii. Application specific

i. **General purpose:**
The General-purpose software is purchased from a software developer and can be used for more than one application.
Examples are word processors, database management systems, and computer aided design systems. They labeled General Purpose because they can solve a variety of users computing problems.

ii. **Application specific:**
Application- specific software can be used only for its intended purpose.
A general ledger system used by accountants and a material requirements planning system used by manufacturing organization are examples of application- specific software.
They can be used only for the task for which they were designed; they cannot be used for other generalized tasks.
Application software can’t be used for general software.

RELATIONSHIP BETWEEN SYSTEM SOFTWARE AND APPLICATION SOFTWARE

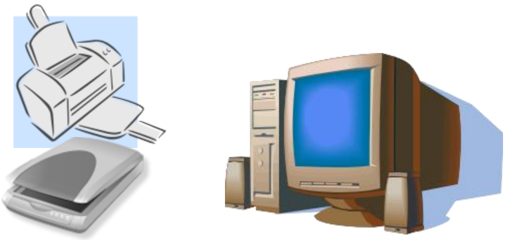


- Each layer represents the connection point user interacts with application software to do some work.
- The application software interacts with operating system, which is a part of system software directly interacts with the hardware.
- The user can directly interact with operating system whenever needed.

COMPUTER ENVIRONMENTS

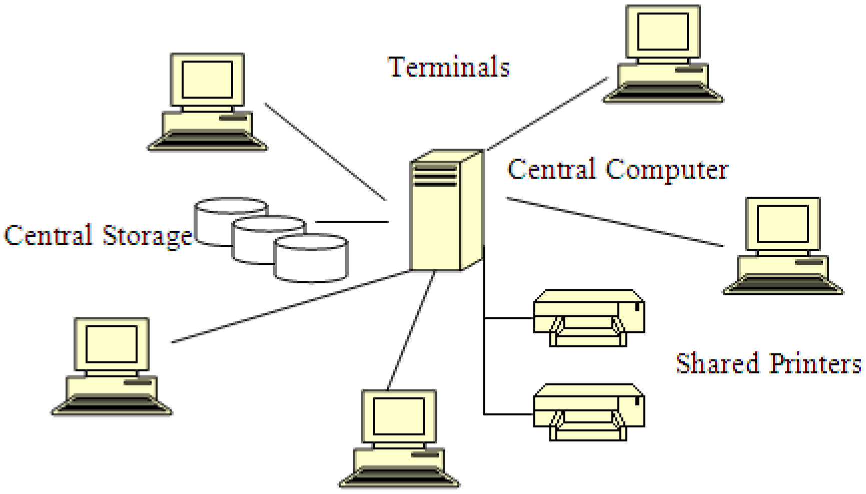
In the earlier days there was only one environment. The mainframe computer hidden in the central computing department with the advent of mini computers and personal computers, the environment changed.

1. PERSONAL COMPUTING ENVIRONMENT



- In this environment, all of the computer hardware components tied together in our PC.
- In this situation, we have the whole computer for ourselves. We can do whatever we want. A typical personal computer is shown above.

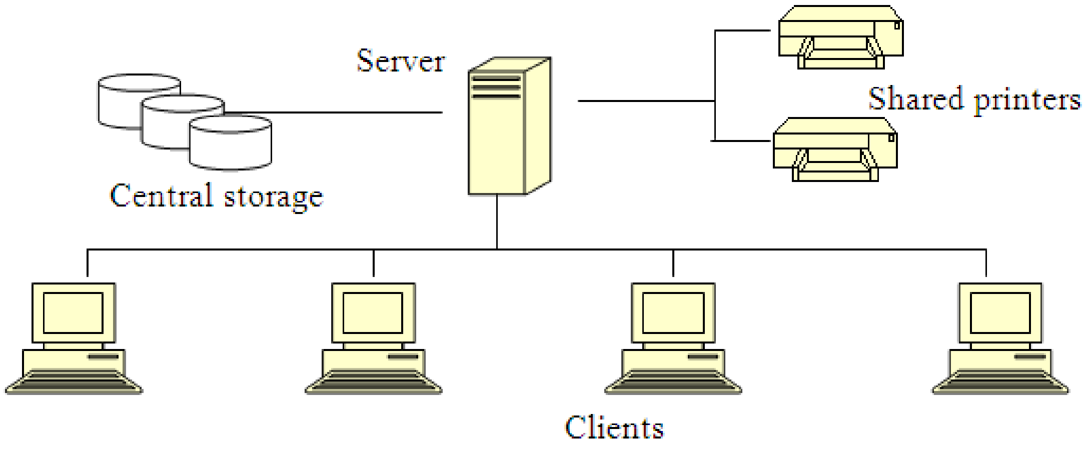
2. TIME SHARING ENVIRONMENT



This type of environment can be found in large companies many users are connected to one or more computers

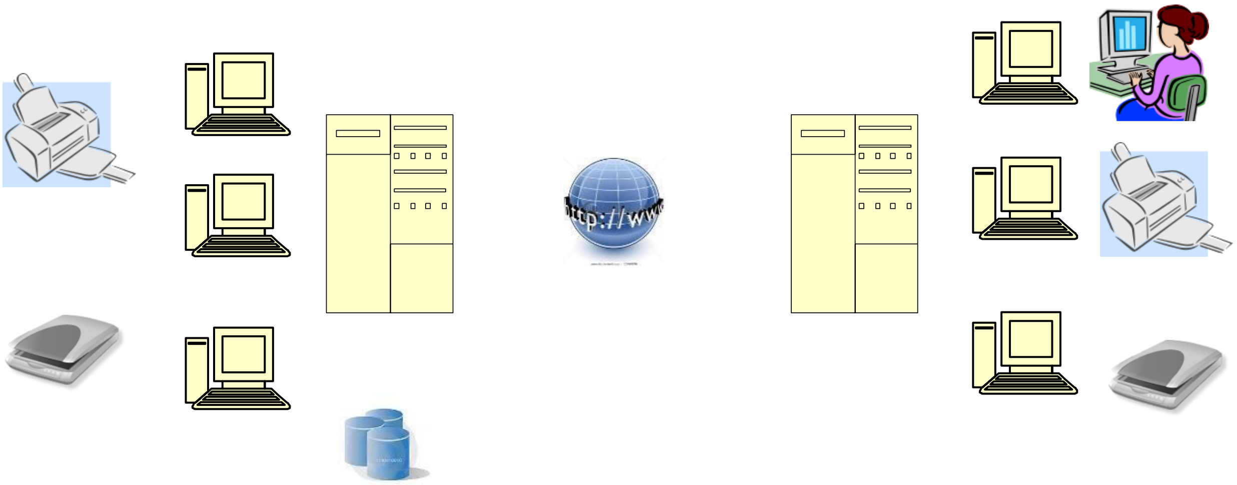
- Each user will be given a terminal.
- All the resources like output devices (such as printers) and auxiliary storage devices (such as discs) are shared.
- The example for time-sharing environment is a college lab in which a minicomputer is shared by many students.
- The central computer has many duties: It must control the shared resources; it must manage the shared data and printing; and it must do the computing. All of this work tends to keep the computer busy. In fact, it is sometimes so busy that the user becomes frustrated by the computer's slow responses.

3. CLIENT SERVER ENVIRONMENT



- In this environment the load of central computer will be shared with different PCs connected to it.
- The users are given PCs or workstations that some of the computational responsibility will be shared.
- Each PC or workstation connected to central computer is called "client".
- The central computer, which may be a powerful microcomputer, minicomputer, or central mainframe system, is known as the server
- A Client/server computing environment splits the computing function between a central computer and users‟ computers

4. DISTRIBUTED COMPUTING:



- A distributed computing environment provides a seamless integration of computing functions between different servers and clients.
- The Internet provides connectivity to different servers throughout the world.
- For example, eBay uses several computers to provide its auction service. This environment provides a reliable, scalable, and highly available network

COMPUTER LANGUAGES

To write a program for a computer, we must use a computer language. Over the years computer languages have evolved from machine languages to natural languages. The computer languages are broadly divided into 3 types.

MACHINE LANGUAGES

- **Machine language** is a collection of binary digits or bits that the computer reads and interprets.
- Machine language is the only language a computer is capable of understanding.
- In this language, instructions are written in 1's and 0's as the computers are made of two state electronic devices that they could understand only pulse that can be in any one of the states: off or on. The off state is represented by 0; the on state is represented by 1. They are called binary code also called as **machine code** or **object code**.
- Computers are identical in design; therefore, each computer has its own machine language.
- Machine languages are usually referred to as "first generation languages/low level languages".
- The exact machine language for a program or action can differ by operating system. The specific operating system dictates how a compiler writes a program or action into machine language.
- Computer programs are written in one or more programming languages, like C++, Java, or Visual Basic. A computer cannot directly understand the programming languages used to create computer programs, so the program code must be compiled. Once a program's code is compiled, the computer can understand it because the program's code is turned into machine language.

Machine language example

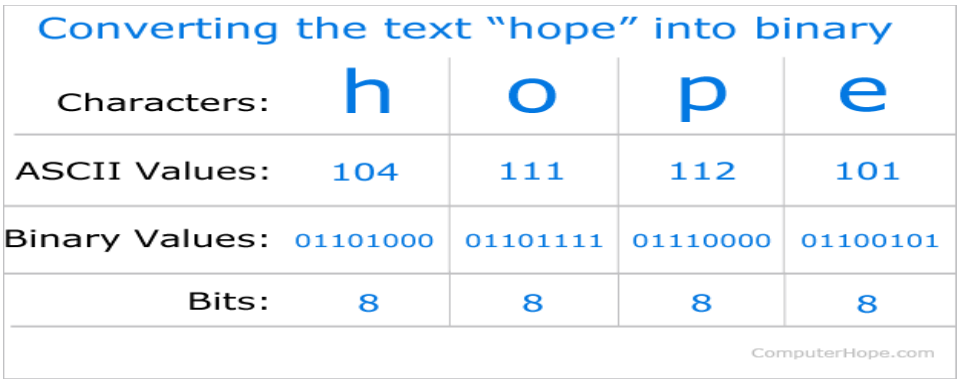
- Below is an example of machine language (binary) for the text "Hello World."

```
 01001000 01100101 01101100 01101100 01101111 00100000 01010111 01101111 01110010 01101100
 01100100
```

- Below is another example of machine language (non-binary), which prints the letter "A" 1000 times to the computer screen.

```
 169 1 160 0 153 0 128 153 0 129 153 130 153 0 131 200 208 241 96
```

- Computers convert text and other data into binary with an assigned ASCII (American Standard Code for Information Interexchange) value. Once the ASCII value is known, that value can be converted to binary. In the following example, we take the word **hope**, and show how it is converted to binary that the computer understands.



Converting the text “hope” into binary

Characters:	h	o	p	e
ASCII Values:	104	111	112	101
Binary Values:	01101000	01101111	01110000	01100101
Bits:	8	8	8	8

ComputerHope.com

- Let us take the first character **h** and break down the process. Once the letter **h** (in lowercase) is typed on the keyboard, it sends a signal to the computer as input. The computer operating system knows the ASCII standard value for **h** is **104**, which can be converted by the computer to the binary value **01101000**.
    - **Bit:** A bit is a value of either a 1 or 0 (on or off).
    - **Nibble:** A nibble is 4 bits.
    - **Byte:** Today, a byte is 8 bits.
      1 character, e.g., "a", is one byte.
    - **Kilobyte (KB):** A kilobyte is 1,024 bytes.
      paragraphs of text.
    - **Megabyte (MB):** A megabyte is 1,048,576 bytes or 1,024 kilobytes.
      873 pages of plain text (1,200 characters).
      4 books (200 pages or 240,000 characters).

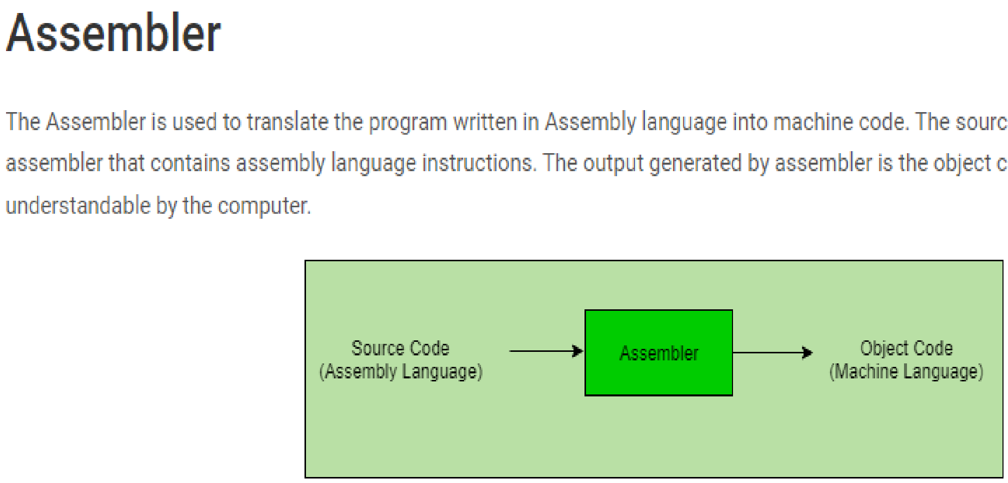
- **Gigabyte (GB)**: A gigabyte is 1,073,741,824 (2^{30}) bytes. 1,024 megabytes, or 1,048,576 kilobytes. **894,784** pages of plain text (1,200 characters).
  4,473 books (200 pages or 240,000 characters).
  640 web pages (with 1.6 MB average file size).
  341 digital pictures (with 3 MB average file size).
  256 MP3 audio files (with 4 MB average file size).
  1 650 MB CD.

- **Terabyte (TB)**: A terabyte is 1,099,511,627,776 (2^{40}) bytes, 1,024 gigabytes, or 1,048,576 megabytes.
  916,259,689 pages of plain text (1,200 characters).
  4,581,298 books (200 pages or 240,000 characters).
  655,360 web pages (with 1.6 MB average file size).
  349,525 digital pictures (with 3 MB average file size).
  262,144 MP3 audio files (with 4 MB average file size).
  1,613 650 MB CDs.
  233 4.38 GB DVDs.
  40 25 GB Blu-ray discs.
- **Petabyte (PB):** A petabyte is 1,125,899,906,842,624 (2^{50}) bytes, 1,024 terabytes, 1,048,576 gigabytes, or 1,073,741,824 megabytes.
  938,249,922,368 pages of plain text (1,200 characters).
  4,691,249,611 books (200 pages or 240,000 characters).
  671,088,640 web pages (with 1.6 MB average file size).
  357,913,941 digital pictures (with 3 MB average file size).
  268,435,456 MP3 audio files (with 4 MB average file size).
  1,651,910 650 MB CDs.
  239,400 4.38 GB DVDs.
  41,943 25 GB Blu-ray discs.
- **Exabyte (EB):** An exabyte is 1,152,921,504,606,846,976 (2^{60}) bytes, 1,024 petabytes, 1,048,576 terabytes, 1,073,741,824 gigabytes, or 1,099,511,627,776 megabytes.
  960,767,920,505,705 pages of plain text (1,200 characters).
  4,803,839,602,528 books (200 pages or 240,000 characters).
  687,194,767,360 web pages (with 1.6 MB average file size).
  366,503,875,925 digital pictures (with 3 MB average file size).
  274,877,906,944 MP3 audio files (with 4 MB average file size).
  1,691,556,350 650 MB CDs.
  245,146,535 4.38 GB DVDs.
  42,949,672 25 GB Blu-ray discs.
- **Zettabyte (ZB):** A zettabyte is 1,180,591,620,717,411,303,424 (2^{70}) bytes, 1,024 exabytes, 1,048,576 petabytes, 1,073,741,824 terabytes, 1,099,511,627,776 gigabytes, or 1,125,899,910,000,000 megabytes.
  983,826,350,597,842,752 pages of plain text (1,200 characters).
  4,919,131,752,989,213 books (200 pages or 240,000 characters).
  703,687,443,750,000 web pages (with 1.6 MB average file size).
  375,299,970,000,000 digital pictures (with 3 MB average file size).
  281,474,977,500,000 MP3 audio files (with 4 MB average file size).
  1,732,153,707,691 650 MB CDs.
  251,030,052,003 4.38 GB DVDs.
  43,980,465,111 25 GB Blu-ray discs.
- **Yottabyte (YB):** A yottabyte is 1,208,925,819,614,629,174,706,176 (2^{80}) bytes, 1,024 zettabytes, 1,048,576 exabytes, 1,073,741,824 petabytes, 1,099,511,627,776 terabytes, 1,125,899,910,000,000 gigabytes, or 1,152,921,500,000,000,000 megabytes.
  1,007,438,183,012,190,978,921 pages of plain text (1,200 characters).
  5,037,190,915,060,954,894 books (200 pages or 240,000 characters).
  720,575,937,500,000,000 web pages (with 1.6 MB average file size).
  84,307,166,666,666,666 digital pictures (with 3 MB average file size).
  288,230,375,000,000,000 MP3 audio files (with 4 MB average file size).
  1,773,725,384,615,384 650 MB CDs.
  257,054,773,251,740 4.38 GB DVDs.
  45,035,996,273,704 25 GB Blu-ray discs.
- Is there anything bigger than a yottabyte?
  As of 2020, there are no approved standard sizes for anything bigger than a yottabyte. However, the two proposed standards are hellabyte or brontobyte.

ASSEMBLY LANGUAGE

- In this language the set of instructions are written in special words like ADD, SUB, INC, HLT so on. These words are called "MNEMONICS" also called "OPCODE" OR"PERATION CODE"
- To convert assembly language to machine code, a tool called “Assembler” is used.
- This is also called “symbolic language”, also referred to second generation programming language, and is also “Low-level language”

```
LOAD BASEPAY
ADD OVERPAY
STORE GROSSPAY
```



Assembler

The Assembler is used to translate the program written in Assembly language into machine code. The source program is a input of assembler that contains assembly language instructions. The output generated by assembler is the object code or machine code understandable by the computer.

{ *Even today, we prefer to use assembly language for*

- *Initialize and test the system hardware prior to booting the Operating System.*
- *This assembly language code will be stored in ROM*
- *Direct interaction with hardware.*
- *In extremely high security situations where complete control over the environment.*
- *Maximize the use of limited resources.*

}

HIGH-LEVEL LANGUAGE

Although symbolic languages greatly improved programming efficiency, they still require programmers to concentrate on the hardware that they were using. And also, it was tedious because machine instruction had to be individually coded. The desire to improve programmer efficiency and to change the focus from the computer to the problem being solved led to the development of high- level language.

- In this language instructions are written in normal English.
- High- level languages are portable to many different computers, allowing the programmer to concentrate on the application problem rather than the intricacies of the computer.
- High- level languages are designed to relieve the programmer from the details of the assembly language.
- High- level languages are too to be converted into machine language. The process of converting them is known as Compilation.
- Some high- level languages are FORTRAN, COBOL, C, C++ etc.
- To convert high level language into machine level language we use a tool called “compiler”.

High-level languages can be classified into 3 categories

- procedure oriented languages (3rdGeneration Languages)
- problem oriented language (4thGeneration Languages)

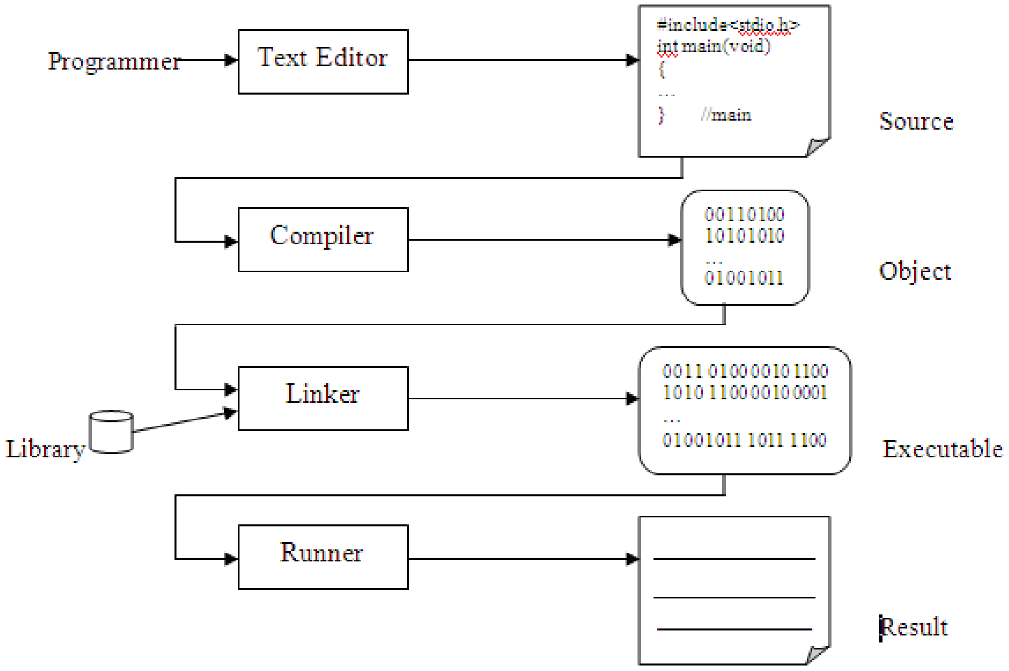
  Ex: To create various items like Text Boxes, Text labels, Radio buttons, etc.
- Natural language (5th Generation Languages)

Ex: LIP, PROLOG used to develop artificial intelligence and expert systems.

CREATING AND RUNNING PROGRAMS

It is the job of a programmer to write and run the program. This process involves four steps.

1. Writing and editing programs
2. Compiling the program
3. Linking the program with required library modules
4. Executing the program



1. WRITING & EDITING PROGRAMS:

- The software used to write program known as a "Text Editor.” A text editor helps us to write, modify and store character data.
- Depending on the editor on our system, we could use it to write letters, create reports, or write programs.
- After we complete the program, we save the file to disk. This file is called **"Source File"**(xxx.c) which is an input to the compiler.
- Our text editor could be a generalized word processor, but it is more often a special editor included with the compiler.
- Some of the features are search commands to locate and replace statements, copy, and paste commands to copy or move statements from one part of a program to another etc.

{The main difference between text processing and program writing is that the programs are written in lines of code, while most text processing is done with, characters and lines. Our text editor could be a generalized word processor, but it is more often a special editor with the compiler.

}

Types of Editors:

There are generally five types of editors as described below:

1. **Line editor**: In this, you can only edit one line at a time or an integral number of lines. You cannot have a free-flowing sequence of characters. It will take care of only one line.
   Ex : Teleprinter, edlin, teco
2. **Stream editors**: In this type of editors, the file is treated as continuous flow or sequence of characters instead of line numbers, which means here you can type paragraphs.
   Ex : Sed editor in UNIX
3. **Screen editors**: In this type of editors, the user can see the cursor on the screen and can make a copy, cut, paste operation easily. It is very easy to use mouse pointer.
   Ex : vi, emacs, Notepad
4. **Word Processor**: Overcoming the limitations of screen editors, it allows one to use some format to insert images, files, videos, use font, size, style features. It majorly focuses on Natural language.
5. **Structure Editor**: Structure editor focuses on programming languages. It provides features to write and edit source code.
   Ex : Netbeans IDE, gEdit.

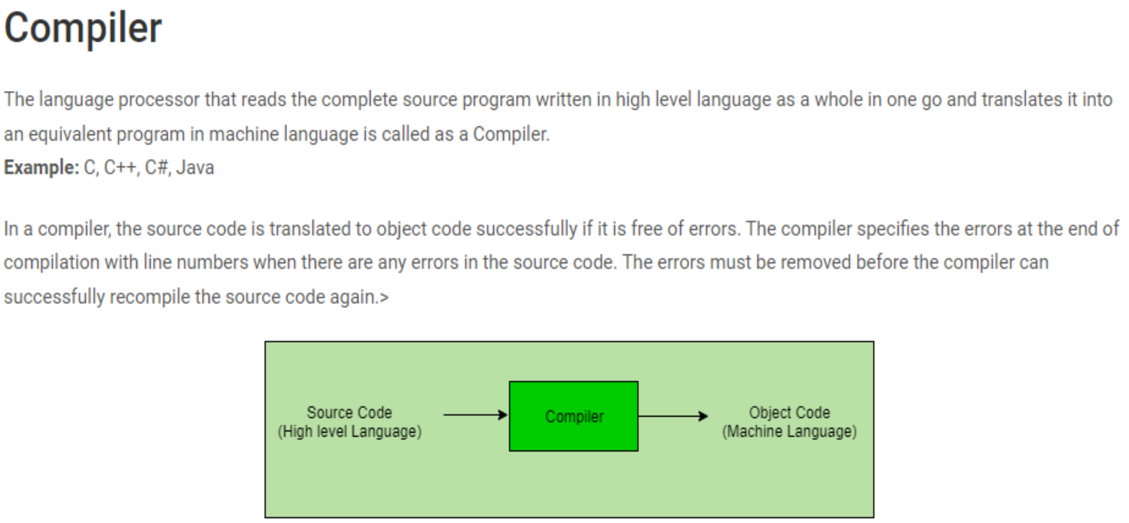
2. COMPILING PROGRAM:

The source code file must be translated into machine language, which is the job of a compiler.

The c compiler has two separate programs.

1. Preprocessor
2. Translator

- The preprocessor reads the source code and prepares it for the translator. While preparing the code, it scans for special instructions known as preprocessor commands.
- These commands tell the preprocessor to look for special code libraries, make substitutions in the code, and in other ways prepare the code for translation into machine language. The result of the preprocessing is called the **Translation Unit**.
- After the preprocessor has prepared the code for compilation, the translator does the actual work of converting the program into machine language. The translator reads the translation unit and writes the resulting object module to a file that can then be combined with other precompiled units to form the final program.
- An object module is the code in machine language called **"Object File"**(xxx.obj file)
- Even though the output of the compiler is machine language code, it is not yet ready to run; that is, it is not yet executable because it does not have the required C and other functions included.



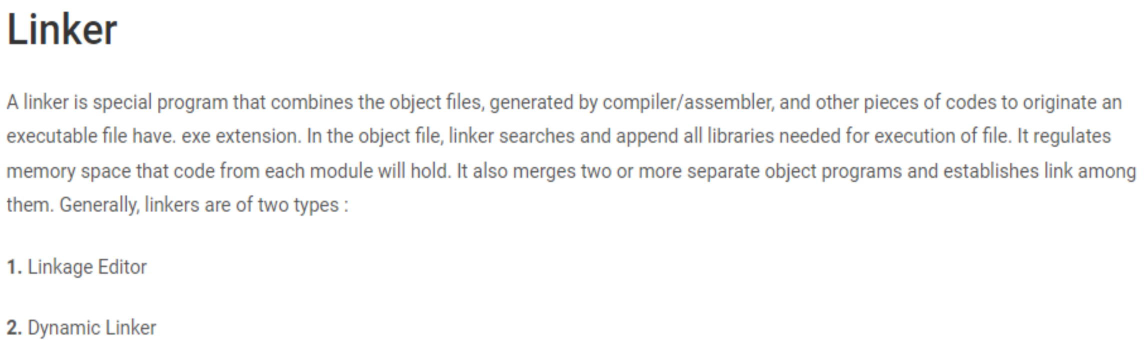
Compiler

The language processor that reads the complete source program written in high level language as a whole in one go and translates it into an equivalent program in machine language is called as a Compiler.
Example: C, C++, C#, Java

In a compiler, the source code is translated to object code successfully if it is free of errors. The compiler specifies the errors at the end of compilation with line numbers when there are any errors in the source code. The errors must be removed before the compiler can successfully recompile the source code again.>

3. LINKING THE PROGRM:

The object file will be linked to the necessary library files such as input/output processes and mathematical library functions exist elsewhere and must be attached to our program using a tool called as linker. After linking, the file is called **"Executable File"**. (xxx.exe file)



Linker

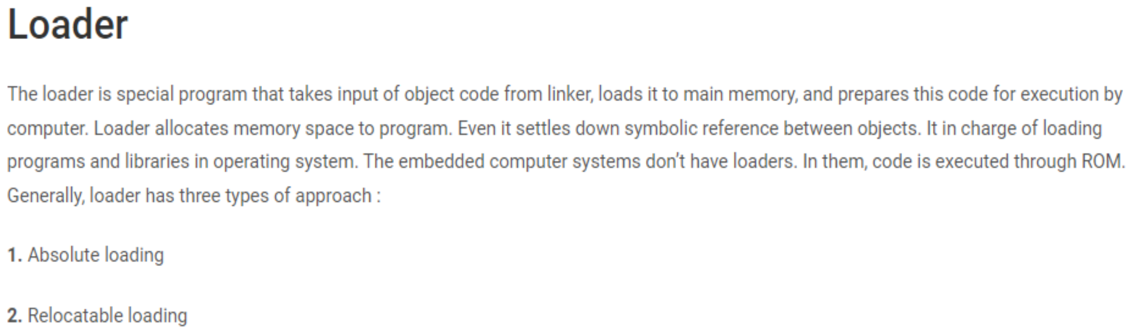
A linker is special program that combines the object files, generated by compiler/assembler, and other pieces of codes to originate an executable file have. exe extension. In the object file, linker searches and append all libraries needed for execution of file. It regulates memory space that code from each module will hold. It also merges two or more separate object programs and establishes link among them. Generally, linkers are of two types :

1. Linkage Editor

2. Dynamic Linker

4. EXECUTING PROGRAMS:

- The executable file, which is in the hard disk, will be loaded in memory for execution by a tool is called “loader.” It locates the executable program and reads it into memory. When everything is loaded, the program takes control and it begins execution.
- In a typical program execution, the program reads data for processing, either from the user or from a file. After the program processes the data, it prepares the output.
- Data output can be to the user’s monitor or to a file. When the program has finished its job, it tells the operating system, which then removes the program from memory.



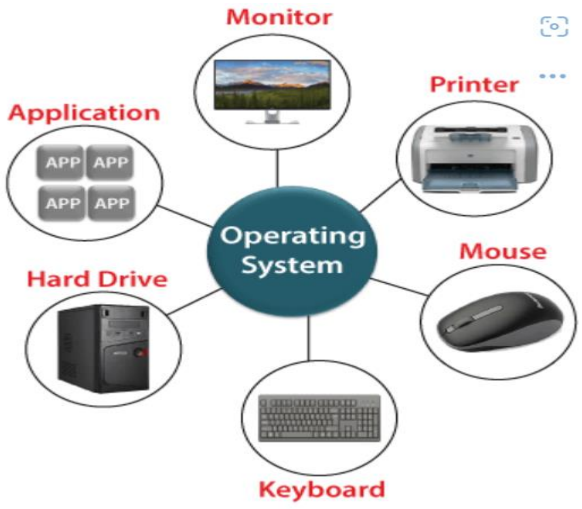
Loader

The loader is special program that takes input of object code from linker, loads it to main memory, and prepares this code for execution by computer. Loader allocates memory space to program. Even it settles down symbolic reference between objects. It in charge of loading programs and libraries in operating system. The embedded computer systems don’t have loaders. In them, code is executed through ROM. Generally, loader has three types of approach :

1. Absolute loading

2. Relocatable loading

Operating System (OS):



As the name suggests, an operating system is a type of software without which you cannot operate or run a computer. It acts as an intermediary or translation system between computer hardware and application programs installed on the computer. In other words, you cannot directly use computer programs with computer hardware without having a medium to establish a connection between them.

Besides this, it is also an intermediary between the computer user and the computer hardware as it provides a standard user interface that you see on your computer screen after you switch on your computer. For example, the Windows and the Mac OS are also operating systems that provide a graphical interface with icons and pictures to enable users to access multiple files and applications simultaneously.

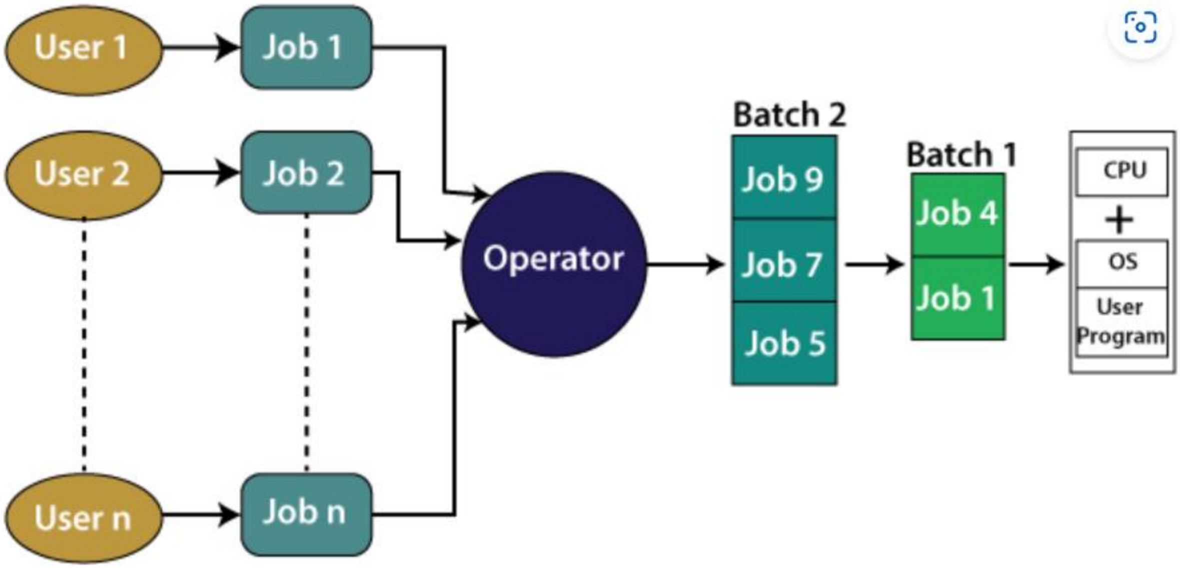
Major Functions of Operating System:

- Memory management
- Processor Management
- Device/ hardware management
- Run software applications
- Data management
- Evaluates the system's health
- Provides user interface
- I/O management
- Security
- Time Management
- Deadlock Prevention
- Interrupt Handling

Types of Operating System:

1) Batch Processing Operating System:

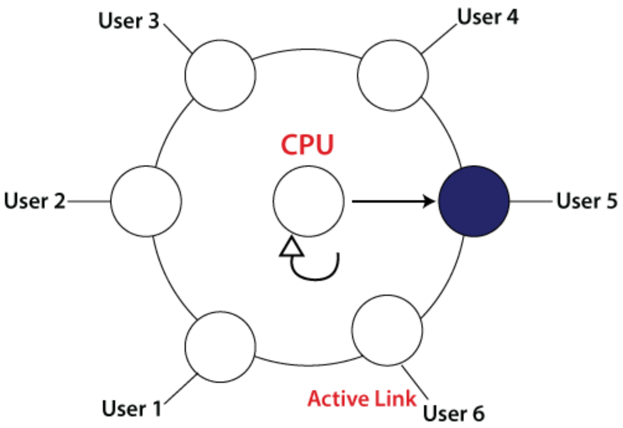
The interaction between a user and the computer does not occur in this system. The user is required to prepare jobs on punch cards in the form of batches and submit them to the computer operator. The computer operator sorts the jobs or programs and keeps similar programs or jobs in the same batch and run as a group to speed up processing. It is designed to execute one job at a time. Jobs are processed on a first-come, first-serve basis, i.e., in the order of their submission without any human intervention.



2) Time Sharing Operating System:

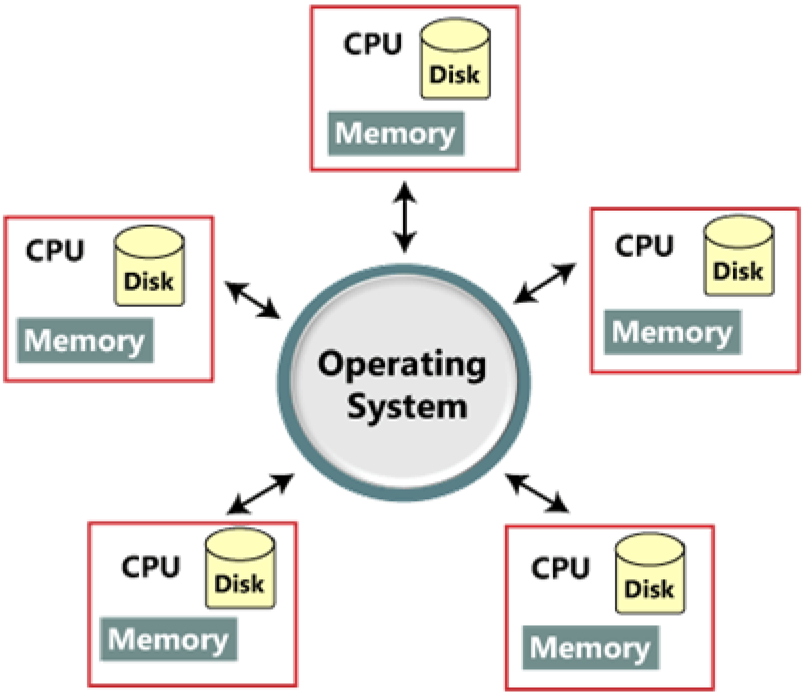
As the name suggests, it enables multiple users located at different terminals to use a computer system and to share the processor's time simultaneously. In other words, each task gets time to get executed, and thus all tasks are executed smoothly.

Each user gets the processor's time as they get while using a single system. The duration of time allocated to a task is called quantum or time slice; when this duration is over, OS starts the next task.



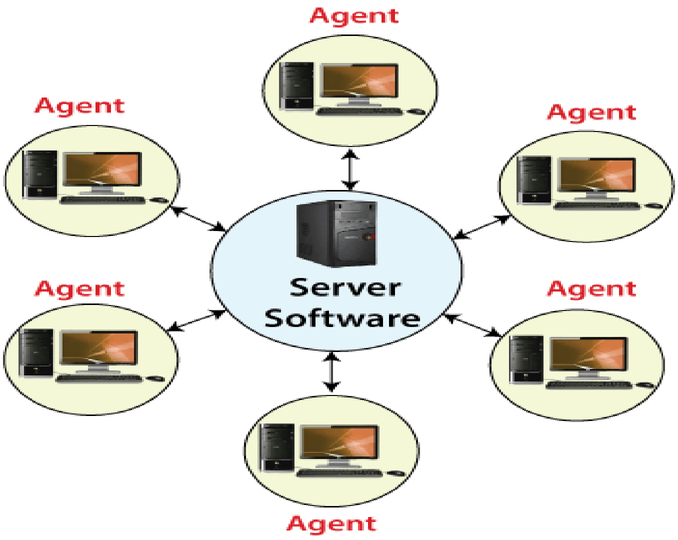
3) Distributed Operating System:

It uses or runs on multiple independent processors (CPUs) to serve multiple users and multiple real-time applications. The communication between processors is established through many communication lines such as telephone lines and high-speed buses. The processors may differ from each other in terms of size and function.



4)Network Operating System:

As the name suggests, this OS connects computers and devices to a local area network and manages network resources. The software in a NOS enables the devices of the network to share resources and communicate with each other. It runs on a server and allows shared access to printers, files, applications, files, and other networking resources and functions over a LAN. Besides this, all users in the network are aware of each other's underlying configuration and individual connections. Examples: Ms Windows Server 2003 and 2008, Linux, UNIX, Novell NetWare, Mac OS X, etc.



5) Real-Time Operating System:

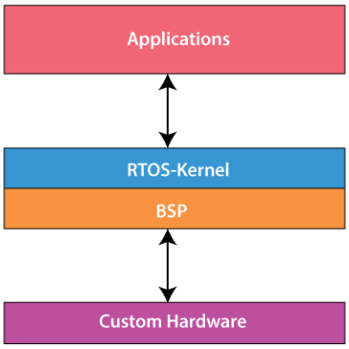
It is developed for real-time applications where data should be processed in a fixed, small duration of time. It is used in an environment where multiple processes are supposed to be accepted and processed in a short time. RTOS requires quick input and immediate response, e.g., in a petroleum refinery, if the temperate gets too high and crosses the threshold value, there should be an immediate response to this situation to avoid the explosion. Similarly, this system is used to control scientific instruments, missile launch systems, traffic lights control systems, air traffic control systems, etc.

Hard Real-Time Systems:

These are used for the applications where timing is critical or response time is a major factor; even a delay of a fraction of the second can result in a disaster. For example, airbags and automatic parachutes that open instantly in case of an accident. Besides this, these systems lack virtual memory.

Soft Real-Time Systems:

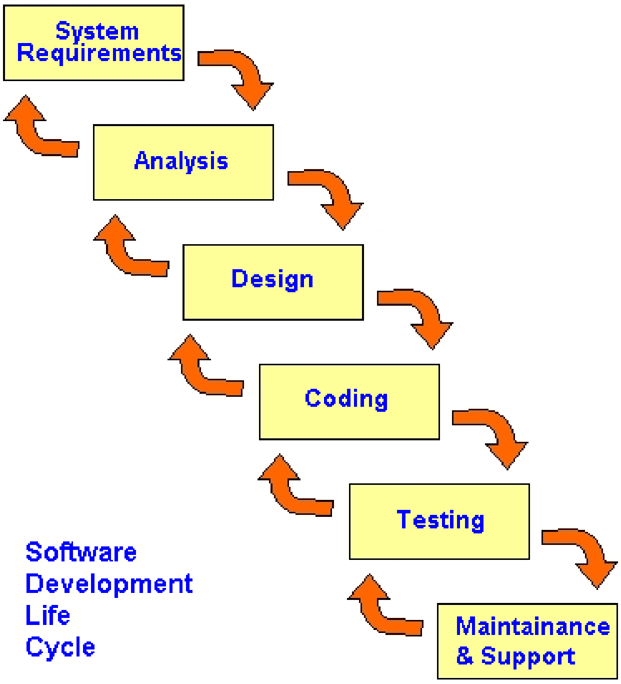
These are used for application where timing or response time is less critical. Here, the failure to meet the deadline may result in a degraded performance instead of a disaster. For example, video surveillance (cctv), video player, virtual reality, etc. Here, the deadlines are not critical for every task every time.



SYSTEM DEVELOPMENT:

System development tells us “The process of overall development of a program that determines the overall quality and success of a program, that carefully enables each problem to be programmed using good structured development techniques, so that our programs will be efficient error free, and easy to maintain.

SOFTWARE (SYSTEM) DEVELOPMENT LIFE CYCLE (SDLC):



Waterfall Model

Today's large scale, modern programming projects built using a series of interrelated phases, which commonly referred as the “SYSTEM DEVELOPMENT LIFE CYCLE”.

It consists of the following phases.

1. In the requirements phase, all the requirements of a project will be gathered, the system analyst define the requirements.
2. In the analysis phase, different alternatives will be analyzed from system's point of view.
3. The design phase determines how the system will be built, the functions of individual programs and the design of files/database is completed.
4. In the coding phase, the programs will be written.
5. In the testing phase, all the programs will be tested together, to make sure that the system works.
6. In the maintenance phase, the project will be put in work into production

PROGRAM DEVELOPMENT:

When we are given the assignment to develop a program, we will be given a program requirements statement and the design of any program interfaces. Also we should receive an overview of the complete project. Our job is to determine how to take the inputs we are given and convert them into the outputs that have been specified. This is known as *program design*.

Program development is a multistep process which consists o the following steps

1. Understand the problem
2. Develop a solution
3. Write a program
4. Test the program

1. Understand the problem:

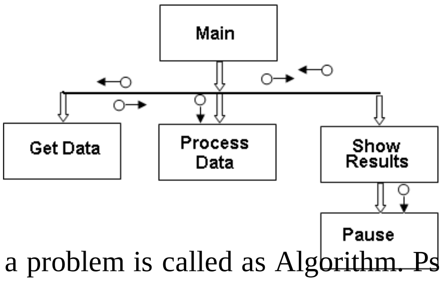
The first step in developing a program is to be carefully understand the given problem. This can be done by asking the questing about the problem to the user.

2. Develop a solution:

Once we finally understand the problem, we use three tools in developing a solution.

i. Structured Chart
ii. Algorithm/Pseudo Code
iii. Flow Chart

i. **A Structured Chart** also known as a “Hierarchy Chart”, is used to design the whole program, which shows the relationship of various units. (The functional flow the program).



Structured Chart

i. **Algorithm**: Step by step procedure to solve a problem is called as Algorithm. Pseudo code: English- like statements that follow a loosely defined syntax and are used to convey the design of an algorithm. Pseudo code is part English, part program logic. Its purpose is to describe, in precise algorithmic detail, what the program being designed is to do. This requires defining the steps to accomplish the task in sufficient detail so that they can be converted into a computer program. Most of the statements in the pseudo code are easy to understand.

ii. **Flowchart:** Graphical representation of an algorithm is called as flowchart. A flowchart is a program design tool in which standard graphical symbols are used to represent the logical flow of data through function.

3. Writing a Program:

After developing a solution, the program has to be written in any language.

4. Test the Program:

After writing the program, it has to be tested to check if there are any errors or not. There are basically two kinds of testing techniques.

i. Black Box Testing
ii. White Box Testing.

- Black Box Testing is done by the testing Engineer and the user.
- These people don't know what is there inside the program.
- They will test the program using the given requirements set.
- White Box Testing done by the programmer who knows what is there inside the program and he can test each line of instruction in a program.

ALGORITHM: (Pseudo Code)

An algorithm is a finite set of instructions that if followed accomplishes a particular task. In addition, algorithms must fallow the below criteria:

1. **Input:** Zero or more quantities are externally supplied.
2. **Output:** At least one quantity is produced
3. **Definiteness:** Each instruction must be clear and unambiguous.
4. **Finiteness:** For all cases of algorithm is should terminate after a finite no of steps.
5. **Effectiveness:** Every instruction must be very basic and easy to fallow

Guidelines for Pseudocode:

- ✓ Algorithms are developed during the *design phase* of software engineering.
- ✓ In the design phase, we first look at the problem, try to write the "*pseudocode*" and move towards the programming (implementation) phase.
- ✓ The logic of a problem can be represented using an algorithm.
- ✓ Algorithm uses English like language.
- ✓ Pseudo code is a high level description of the algorithm
- ✓ It is less detailed than the program
- ✓ It will not reveal the design issues of the program

Algorithms are divided into three categories

i. Sequence
ii. Selection
iii. Iteration

i. Sequence:

In this category all the instructions are performed one after the other.

Ex: Write an algorithm for telephone about the conversation between two people. Step 1: Dial the Number.
Step 2: Phone rings at the called party Step 3:
Caller waits for the response.
Step 4: Phone lifted by the called party. (Connection Activated). Step 5:
Conversation begins.
Step 6: Release the connection. (Connection Terminated)

ii. Selection:

In the selection, instructions will be executed based on the condition.

□ The selection form can be written as

```
        if (condition) then
              stateme
        nt;
if (condition) then
```

```
                        statement1;
                else
                        statement2;
```

Ex: Take the previous example.

```
   Step 1: Dial the Number
                if   (busy   tone)
                then goto step1;
   Step 2: Phone rings at the called party
   Step 3: Caller waits for the response.
   Step 4: Phone lifted by the called party. (Connection Activated). Step 5:
   Conversation begins.
   Step 6: Release the connection. (Connection Terminated)
```

iii. Iteration:

In this category some of the instructions will be performed repeatedly for some number of times. Iteration statements are written using the following format.

```
         Repeat
                Statements;
         Until
                Condition
```

- The sys will be executed continuously if the condition is false
- If the condition becomes true the next step will be executed

Algorithm1:

Write a program for adding 2 numbers

sol:

```
Algorithm                              Pseudo code:
step 1: start                        step     1:          start
      2: read 2 numbers                    2:             read 2 numbers a,b
      3:add the 2 numbers                  3:              sum<-a+b (or) sum=a+b;
      4:print sum                          4:              Display sum
      5:stop                               5:              Stop
```

Algorithm 2:

Write an algorithm for finding the average of 3 numbers

sol:

```
step 1: start
    2: read 3 numbers a, b, c
    3: avg<- a+b+c/3.0
    4: print avg
    5: stop
```

Algorithm 3:

Write an algorithm to find the area of triangle

Sol:

```
Step 1:  start
     2: Read 2 numbers b, h
     3: Area<-(b*h)/2.0
     4:   print area
     5: stop
```

Algorithm 4:

Write an algorithm to find the area of circle

Sol:

```
Step    1:   start
        2:   read radius "r";
        3:   Area <-(22/7.0)*r*r (or) 3.14 (r*r*)
        4:   print area;
        5:   stop
```

Algorithm 5:

Write an algorithm to find the given number is even or odd

Sol:

```
STEP1:       start
        2:   Read number 'n'
        3:   if (n==0) then print "Number is neither even nor odd;
        4:   if (n%2==0) then print "Number is even ";
        If (n%2==0) then print "Number is odd ";
        5:   stop;
```

```
STEP1:       start
        2:   Read number 'n'
        3:   if (n==0) then
        print "Number is neither even nor odd;
       4:    else
                if (n%2==0) then
                        print "Number is even ";
               else
                        print “number is odd”
       5:stop
```

Algorithm 6:

Write an algorithm to find whether a number is +ve or not

Sol:

```
STEP 1:   Start
     2:   Read a number N
     3: if (n=0) print “N is neither +ve  nor -ve”
      4:  else if (n>0)
        print” n is +ve”
           else
         Print” n is -ve”
     5: Stop
```

Algorithm 7:

Write a number to find the biggest of the two numbers

Sol:

```
STEP 1:   start
        2:  read two numbers;
        3:  if (a=b) then
        Print “both are same” else
        if (a>b) then
        Print “a is bigger than
        b”
            Else
            Print “b is bigger than a”
       4: stop.
```

Algorithm 8:

Write an algorithm to swap given two numbers 10,20.

Sol:

```
STEP 1:   start
        2:  read
        a<-10;
        b<-20;
        3:  c<-a;
           a<-b;
           b<-c;
     4:print a, b;
    5:  stop.
```

Algorithm 9:

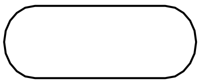
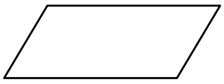
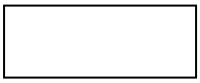
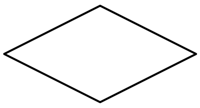
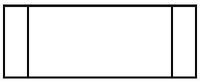
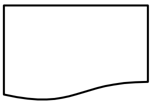
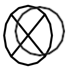
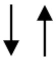
Write an algorithm to swap two numbers without using third name(variable).

Sol:

```
STEP 1:   start
        2:  read a←10;
                b←20
        3: a← a+b;
           b←a-b;
           a←a-b;
        4: print a, b;
        5: stop
```

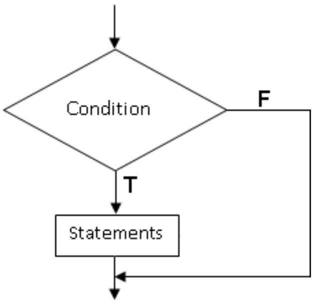
FLOW CHART:

A Flow chart is a diagrammatic representation of the given problem. The different symbols used for a flow chart given below.

SYMBOL	OPERATION	DESCRIPTION
	Start/Stop	**Terminator:** Shows the starting and ending points of the program. A terminator has flowlines in only one direction, either in (a stop node) or out (a start node).
	Data Input/ output given Statements	This represents the Input / Output of a problem
	Process	**Processing:** Indicates an operation performed by the computer, such as a variable assignment or mathematical operation.
	Decision	**Decision:** The diamond indicates a decision structure. A diamond always has two flow lines out. One flow line out is labeled the “yes” branch and the other is labeled the “no” branch.
	Predefined process	**Predefined Process**: One statement denotes a group of previously defined statements. For instance, “Calculate m!” indicates that the program executes the necessary commands to compute m factorial.
	Document	This represents the situation to produce a document as a result, or for comment a particular situation
	Connector	This symbol used to continue the flow chart
	Summing junction	This is used for connecting multiple flow
	lines	
	Flow lines	These symbols represent the flow of the execution of a given problem

The conditional symbol can be used in different forms.

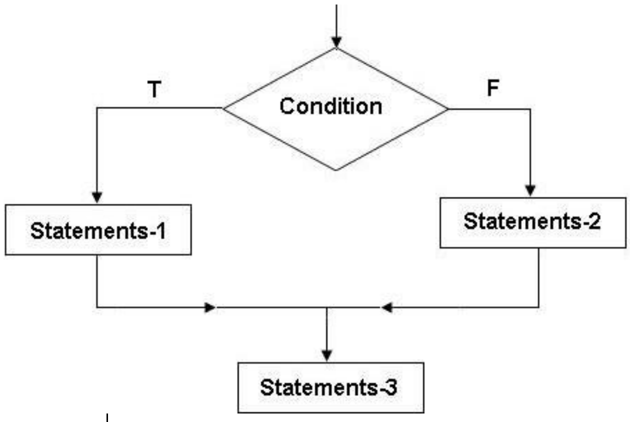
1) Single alternative Decision



- If the condition is true the statements will be executed and goes to the next instructions.
- If the condition is false the control goes directly to the next step.

2)Two alternative Decisions

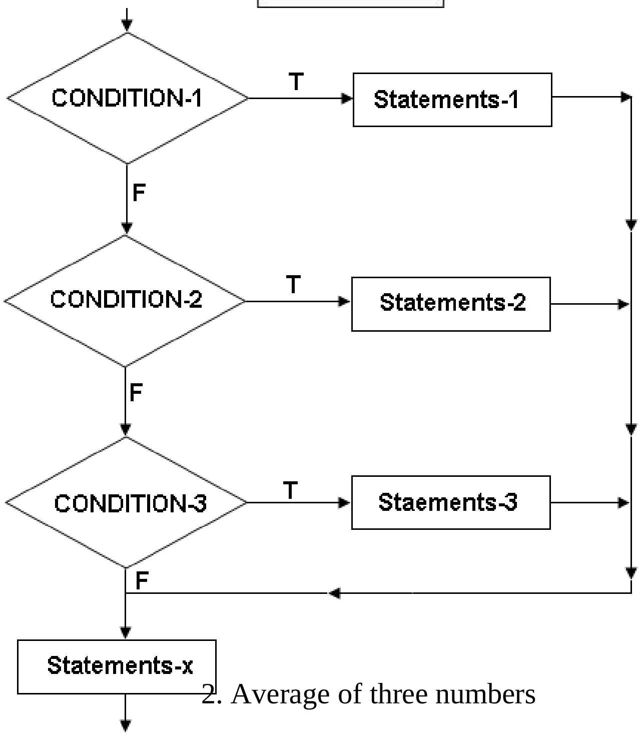
If the condition is true then stts-1 will be executed and control goes to the next instruction, else i.e., the condition is false then the stts-2 will be executed and control goes to the next instruction.



3) Multi alternative decisions

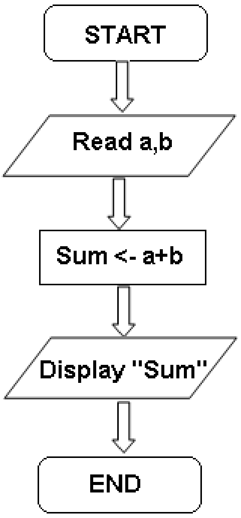
If the condition -1 is true, stts-1 will be executed and control goes to the next instruction, if condition-1, is false then condition-2 will be checked

If the condition-2 is true, stts-2 will be executed and goes to the next instruction else, the condition-3 will be checked and this will go on...

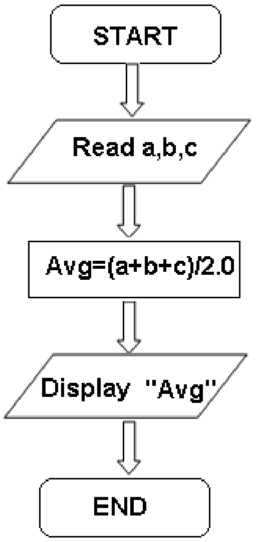


(1) Draw flow charts for the following: -

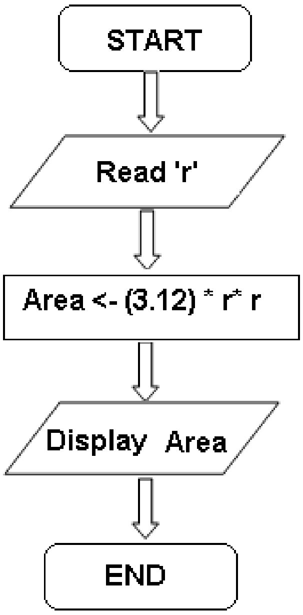
1. Addition of two numbers



2. Average of three numbers

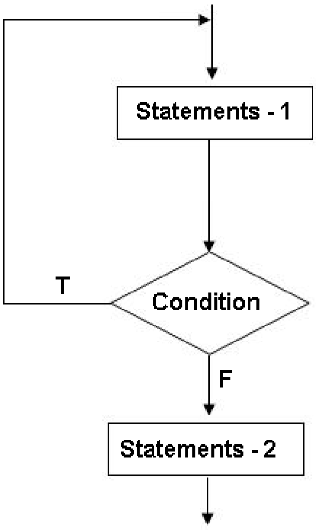


3. Area of a circle



The iteration can be specified in three different forms: - i). Repeat-Until

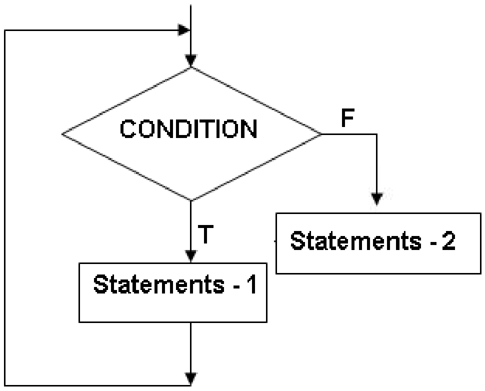
```
Repeat
        Statements
– 1 Until
        Condition;

Statements – 2
```



ii). While form

```
While(cond-1)
 Statements-1
 End
 While staements-2
```

In this stt-1 will be repeated if the condition is true. if it is false, the control go to next statement, statements-2.



iii).

do-while Statement

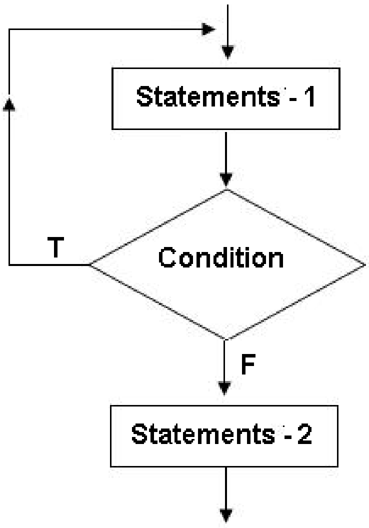
do

Statements – 1;

while (condition)

Statements – 2;

In this statements-1 will be repeated if the condition is true.
If it is false the flow (control)goes to the next statements-2.



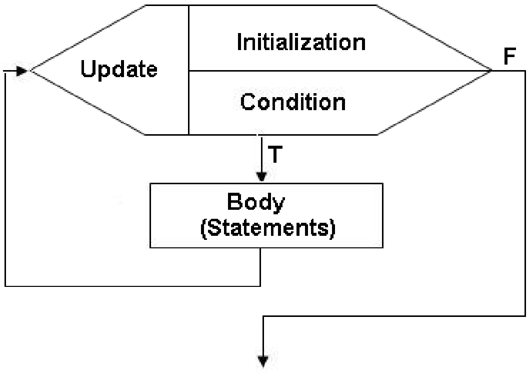
***

The difference between while and do while is do while statements will be executed at least once without checking the condition.

iv.) for loop:

In for loop, a particular variable will be initialized with a value, and then the condition will be checked, if it is true, the body of the statements will be executed.

After that, the initialized value will be updated, and again control checks the condition, if it is true, the process continues, else the control comes out of the loop.



PRACTICE:

1) Draw a flowchart for finding the sum of „n‟ numbers starting from 1?
2) Write an algorithm to read five integers and find out if the values are in ascending order.
3) Draw a flow chart for displaying the grade of the marks of average of 4 subjects of a student.

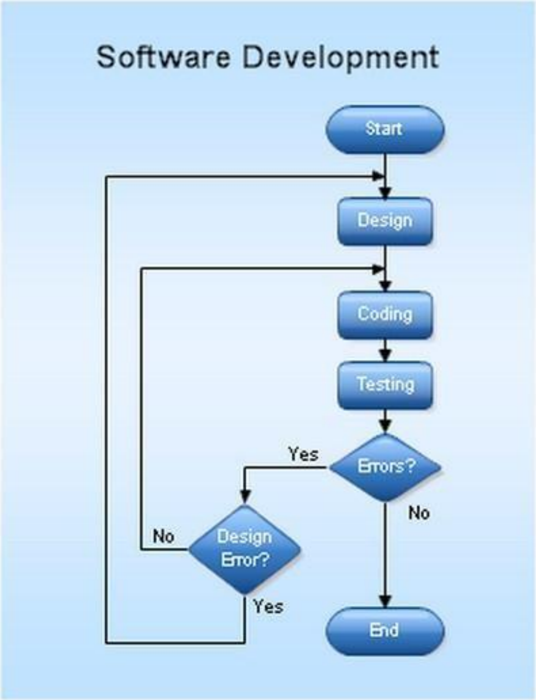
Average	Grade
>=75	A
>=60 &&<75	B
>=50 &&<60	C
<50	D

** Ensure that the user entered all of his marks in between 0 & 100 only.

4) Draw a flow chart to read ten integer values and print the sum of squares of the values.
5) Calculate the Expenditure per month of a sales person. Your algorithm should print either „dues‟ or „no dues. If the expenditure is below 5000/- then the balance(5000-exp) is his due. If the expenditure is above Rs. 5000/- then no Due.
   Hint: subtract exp from 5000 if bal is <5000 the print the due else print no dues

EX1.System Development Life Cycle:

In this example flow chart, I have checked the condition regarding the design errors only, students can check the errors from coding phase to requirements phase also, in which we have discussed in previous section.


Software Development
Start
Design
Coding
Testing
Errors?
Yes
No
No
Design Error?
Yes
End